

**Project Spot Check:**

**Accuracy Enhancement of Software to Aid in Recognition of Snow Leopards and Containerizing the Code**

**ECE 20.4’s User Guide to Azure Databricks version of Recognition.py**





Contents

[Table of Figures 2](#_Toc41307025)

[Introduction 3](#_Toc41307026)

[Azure Resource Setup 3](#_Toc41307027)

[Storage Account 4](#_Toc41307028)

[Configuring the Working Directory 5](#_Toc41307029)

[Enterprise Application 6](#_Toc41307030)

[Azure Databricks Workspace 9](#_Toc41307031)

[Running the Recognition Spark Notebook in Databricks 10](#_Toc41307032)

[Recognition Spark Notebook Output 13](#_Toc41307033)

[Progress of the Recognition Notebook 14](#_Toc41307034)

[References 15](#_Toc41307035)

# Table of Figures

[Figure 1. Limited Quota when using Azure for Students subscription 3](#_Toc41307036)

[Figure 2. Azure Portal landing screen 4](#_Toc41307037)

[Figure 3. Storage account configuration 4](#_Toc41307038)

[Figure 4. Container creation 5](#_Toc41307039)

[Figure 5. Necessary directories and files in Recognition working directory, highlighted 5](#_Toc41307040)

[Figure 6. Selecting the snow leopard images to upload once in the created "images" directory 6](#_Toc41307041)

[Figure 7. Working directory configuration 6](#_Toc41307042)

[Figure 8. Add an enterprise application 6](#_Toc41307043)

[Figure 9. Application ID from enterprise application recorded in Notepad 7](#_Toc41307044)

[Figure 10. Secret generated and recorded in Notepad 7](#_Toc41307045)

[Figure 11. Directory ID recorded in Notepad 8](#_Toc41307046)

[Figure 12. Access control location in storage account overview 8](#_Toc41307047)

[Figure 13. Assignment of Storage Blob Data Contributor role to enterprise application 9](#_Toc41307048)

[Figure 14. "Launch Workspace" icon 9](#_Toc41307049)

[Figure 15. Databricks landing page, with "Clusters" highlighted 10](#_Toc41307050)

[Figure 16. Cluster configuration for quick\_set tests 10](#_Toc41307051)

[Figure 17. Libraries installed on the newly-made cluster 11](#_Toc41307052)

[Figure 18. Notebook creation from launch page 11](#_Toc41307053)

[Figure 19. First command in the notebook, which takes care of mounting with credentials collected from the Azure Portal 12](#_Toc41307054)

[Figure 20. Path strings that are edited if the mount point is edited 12](#_Toc41307055)

[Figure 21. Termination of Databricks cluster, with the "State" and the "stop" icon highlighted 13](#_Toc41307056)

[Figure 22. Score matrix output displayed in Databricks 13](#_Toc41307057)

[Figure 23. Score matrix accessed in storage container through the Azure Portal 14](#_Toc41307058)

# Introduction

One of the objectives specified by Panthera to team ECE 20.4 was to perform testing on different sets of spotted cat images and large data sets. When the team was introduced to Recognition in fall 2019, the Python program was shown to have a bottlenecking issue. On a local machine with datasets larger than 25 images, Recognition took more than half an hour or up to 3 hours to perform keypoint matching output a score matrix.

As part of the team’s design for having a containerized, portable version of Recognition that works on multiple operating systems, ECE 20.4 chose to use Azure Databricks for cloud computing. This would allow multiple “worker” machines to be called to perform computations for matching and the score matrix, decreasing total runtime when working with large datasets. The result of ECE 20.4’s work was a Python notebook which can be pasted into a session in Databricks. Databricks is also offered as a service in Amazon Web Services, which is why this notebook was chosen as a deliverable to Panthera.

The purpose of this guide is to provide instructions and guidance to future development teams for running the Recognition notebook using Databricks in Azure. It will cover creation of a Recognition working directory in an Azure storage container, creating an enterprise application for accessing the storage container through Databricks, and creation and usage of an Azure Databricks Service for running the notebook. This is meant to save time searching the official Azure documentation (<https://docs.microsoft.com/en-us/azure/>) and Databricks documentation (<https://docs.databricks.com/> or <https://docs.microsoft.com/en-us/azure/azure-databricks/>), which can be used alongside these instructions. The progress and function of Recognition as a Spark notebook, as well as recommendations for further development, are included at the end of this document.

# Azure Resource Setup

This section will guide the user through creating the storage account, enterprise application, and Azure Databricks Service. These instructions assume the user has already created a “Default Directory” in Azure by signing up for a Pay-As-You-Go account in Azure. ECE 20.4 developed in Azure because the Seattle University ECE Department offers $100 in credit to its students. However, the team learned in winter 2020 that these student accounts had a low ceiling for their quota; no more than 4 Standard DSv2 Family vCPUs could be requested.

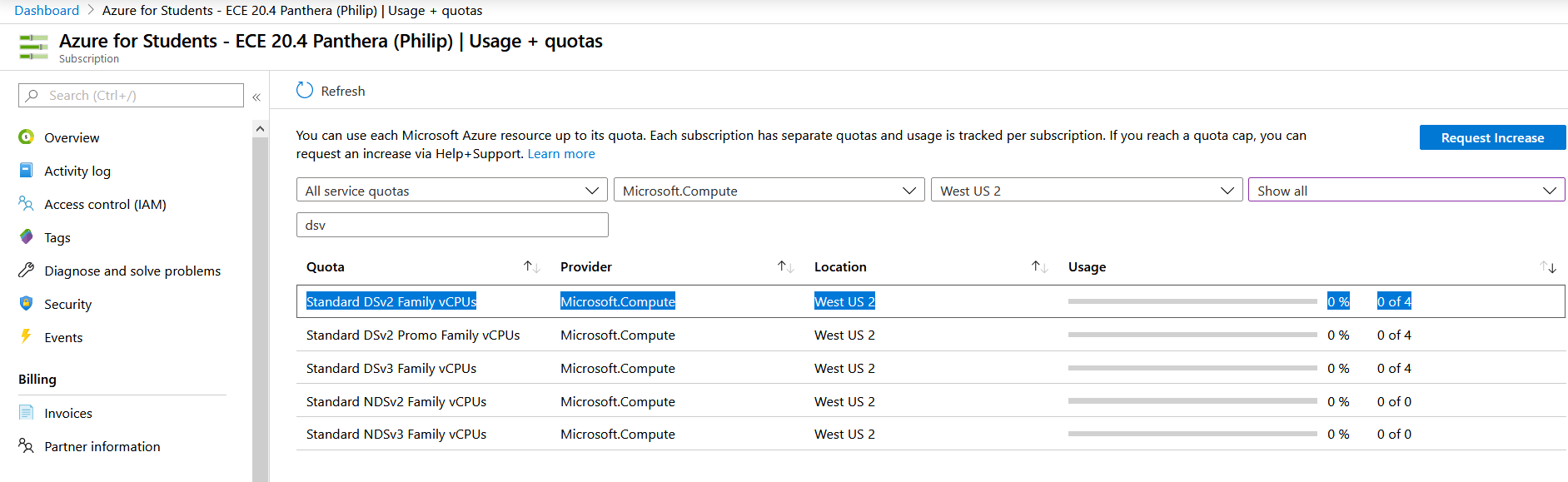


Figure 1. Limited Quota when using Azure for Students subscription

## Storage Account

To begin, a storage account and container will need to be created for holding the images and templates used by Recognition. Upon creating the Azure subscription, the user will see the following screen, accessible by going to <https://portal.azure.com>:

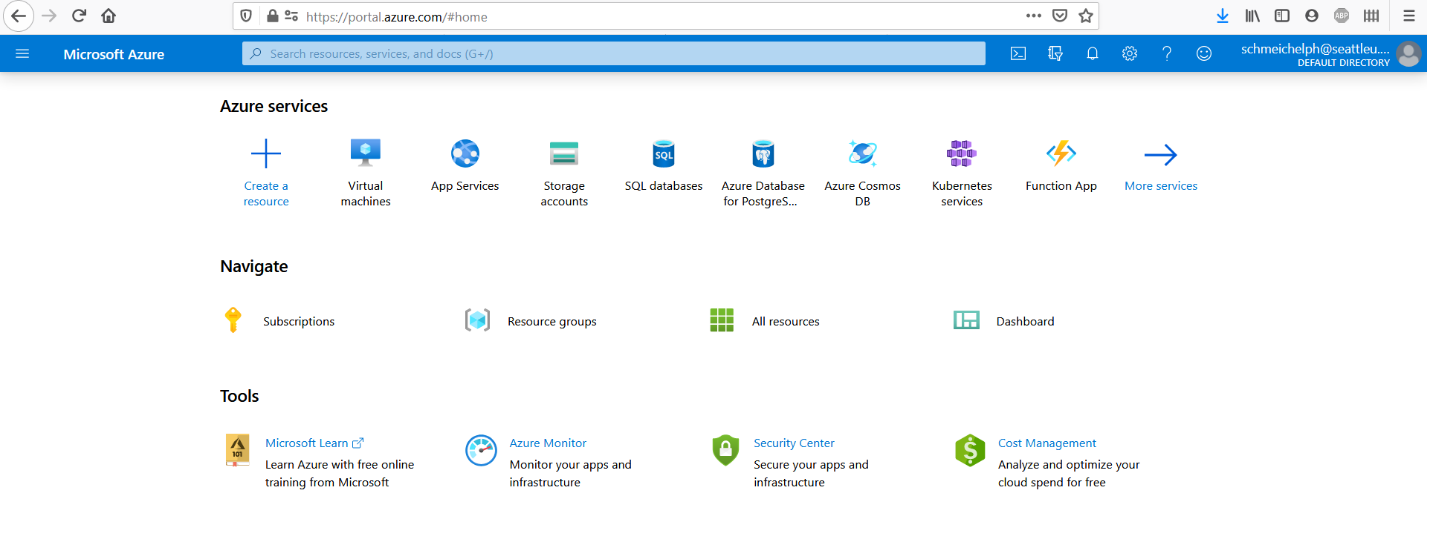


Figure 2. Azure Portal landing screen

By searching or clicking either “Create a resource” or “Storage accounts,” you can access your subscription’s storage accounts page. From here, you can click “Add” and create the storage account. As shown in the below figure, you may need to create a new resource group, and a name is required for the storage account. It is important to proceed to “Advanced” and select “Hierarchical namespace: Enabled” before hitting “Review + create.” This is because of the working directory (file-folder system) that Recognition is designed to use.

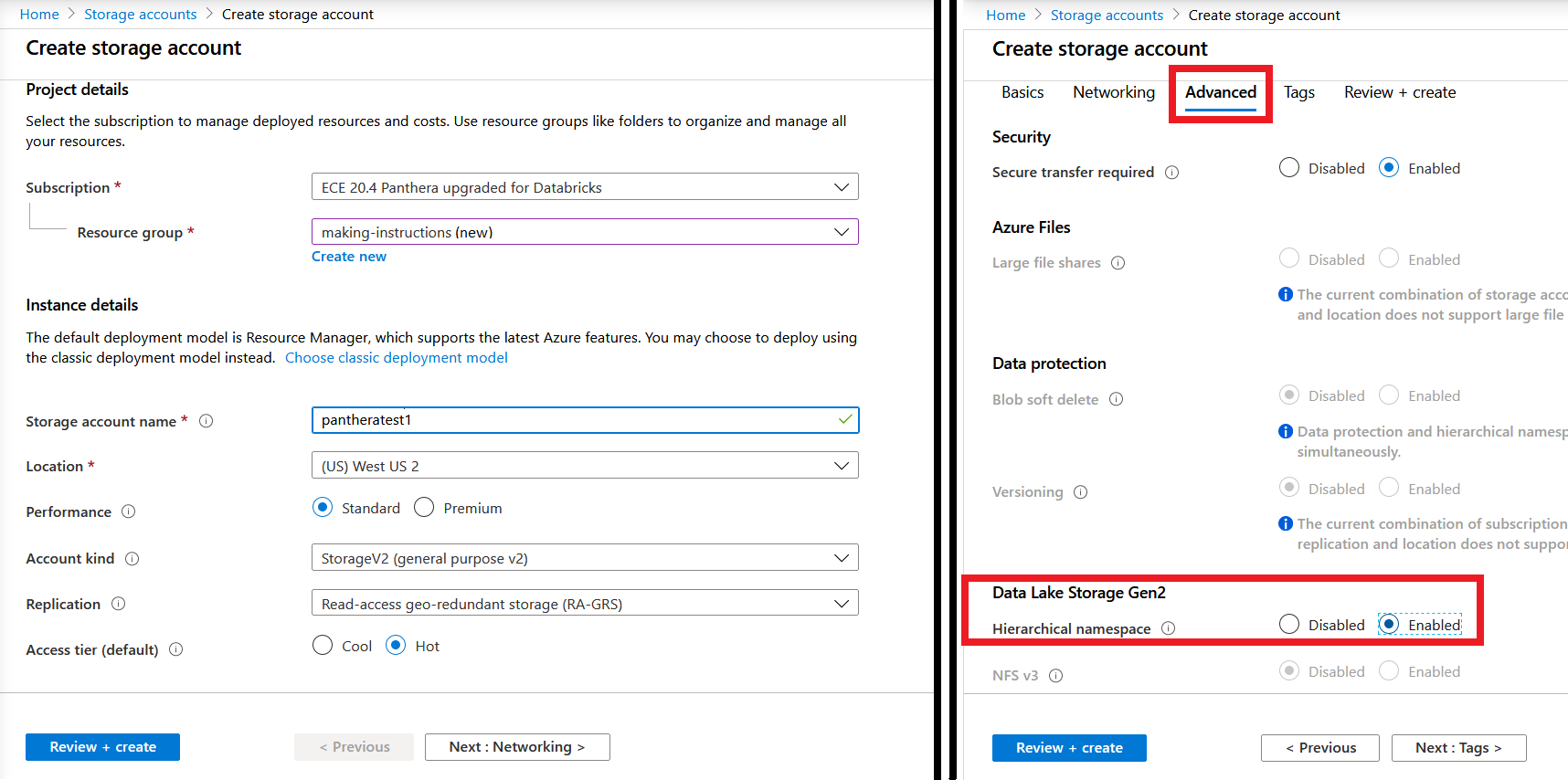


Figure 3. Storage account configuration

When the storage account is created, access it using “Go to resource” or by searching for “Storage accounts” in the Azure Portal. In the overview, select “Containers” and create a container, ensuring that “Container (anonymous read access for containers and blobs)” is selected.

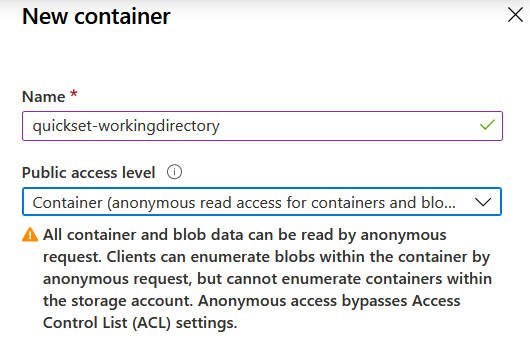


Figure 4. Container creation

## Configuring the Working Directory

When the container is created, you will need to insert the images, templates, and *config.json* file as they are given in a regular Recognition working directory. In the container you just created, use “Add directory” to create the “images” and “templates” directories and use “Upload” to upload the *config.json* file.

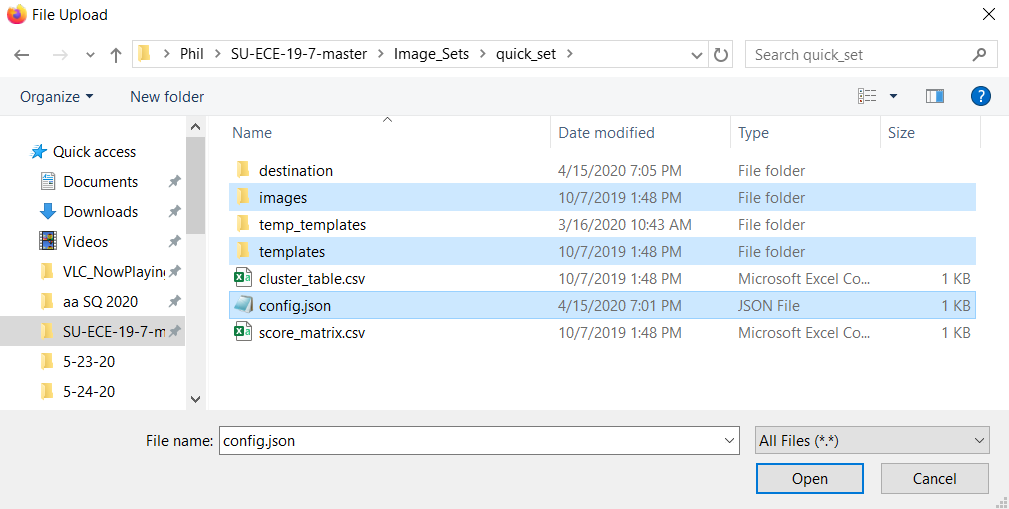


Figure 5. Necessary directories and files in Recognition working directory, highlighted

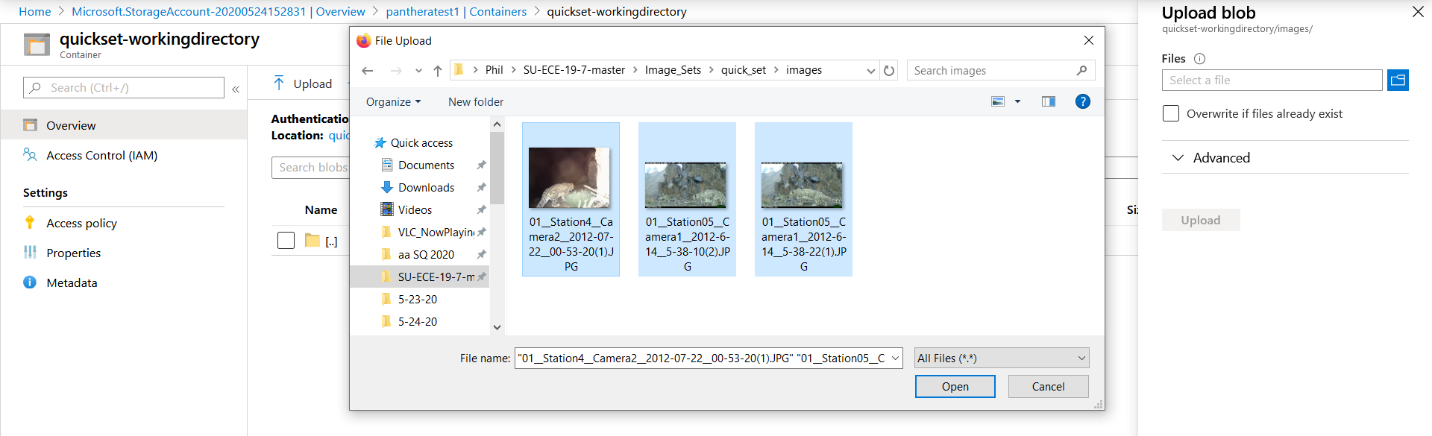


Figure 6. Selecting the snow leopard images to upload once in the created "images" directory

The images and templates should share the same names, except for the extensions (where images are “.JPG” and templates are “.BMP”). When finished, the container should look as shown in the figure below in Azure.

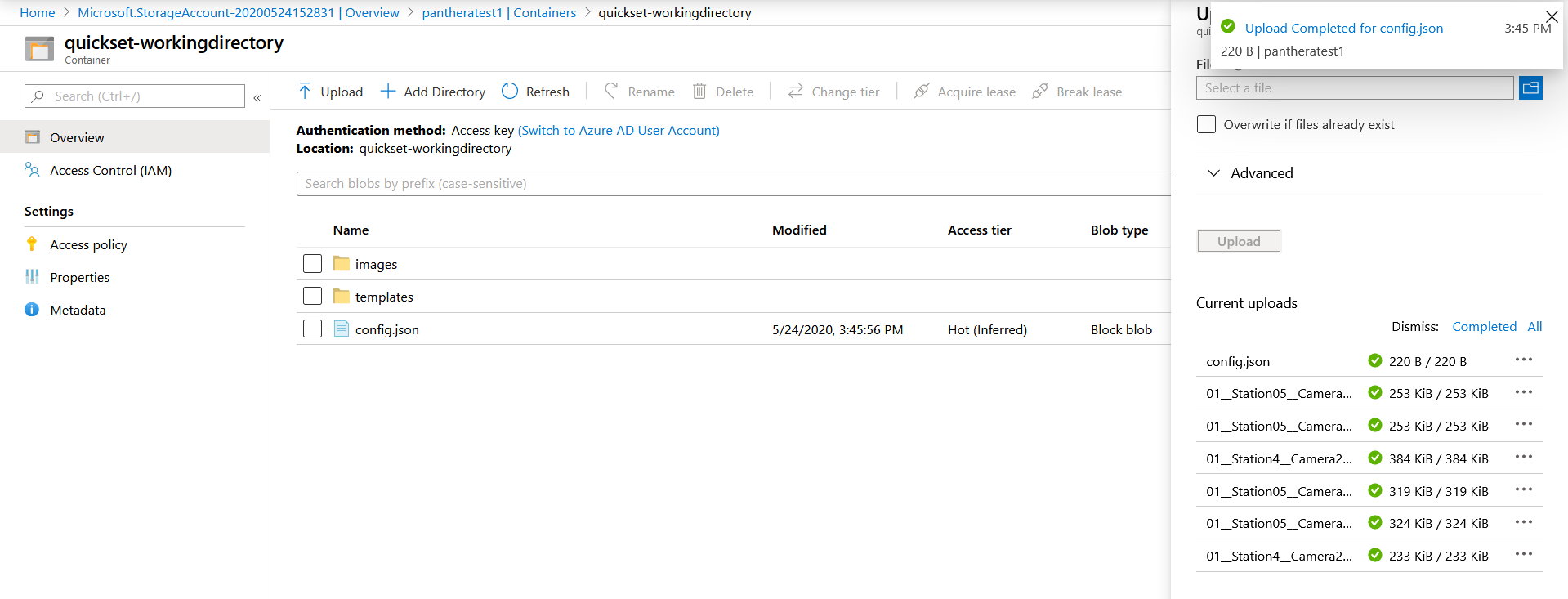


Figure 7. Working directory configuration

## Enterprise Application

An enterprise application must be created to give Databricks a user that can access the storage account through Databricks’ required credentials, when it comes time to mount. In the Azure Portal, search for “Enterprise applications.” Once there, “Add an application” and select “Application you’re developing.”

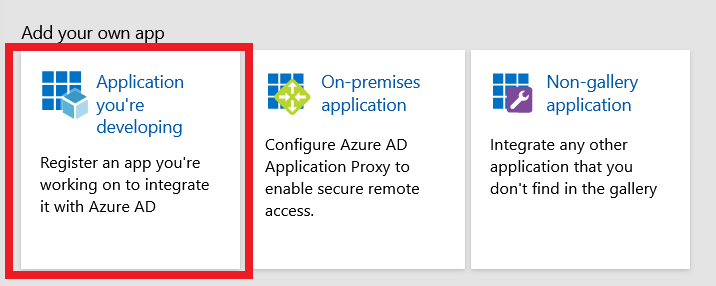


Figure 8. Add an enterprise application

This will then take you to “App registrations,” where you will create a “New registration.” Now, you will need to record credentials that will be used in the Spark notebook for running Recognition. Record the “Application (client) ID” shown on the Overview page of your new enterprise application. Then, go to “Certificates & secrets” and select “New client secret” to generate and record the secret.

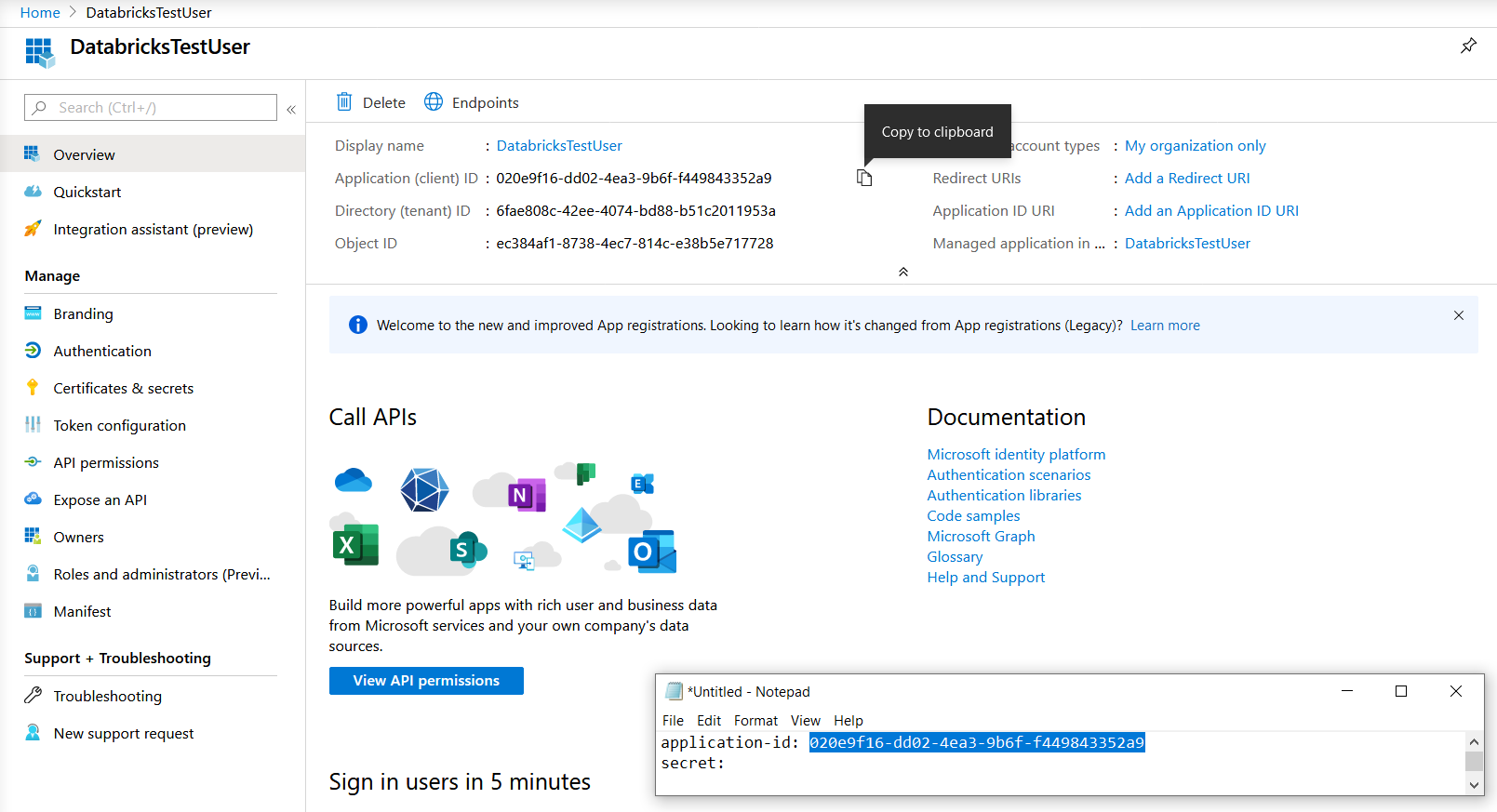


Figure 9. Application ID from enterprise application recorded in Notepad

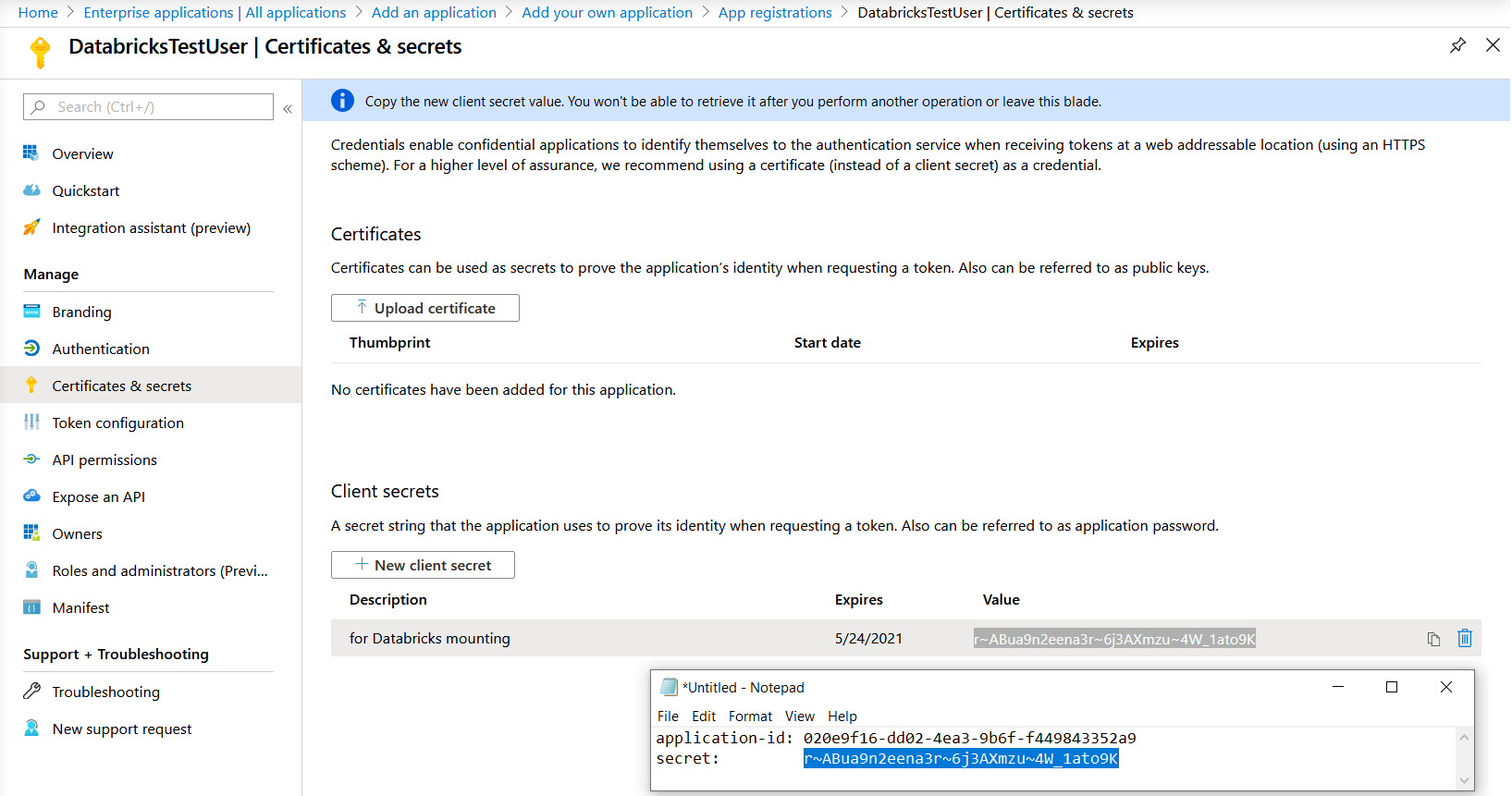


Figure 10. Secret generated and recorded in Notepad

The final code to be copied and used later is the Directory ID, which can be accessed anytime in the Azure Portal by clicking the notebook-and-funnel icon in the top-right. Copy the code underneath ensuring that the code is for this Default Directory associated with the Pay-As-You-Go account where you created these Azure resources.

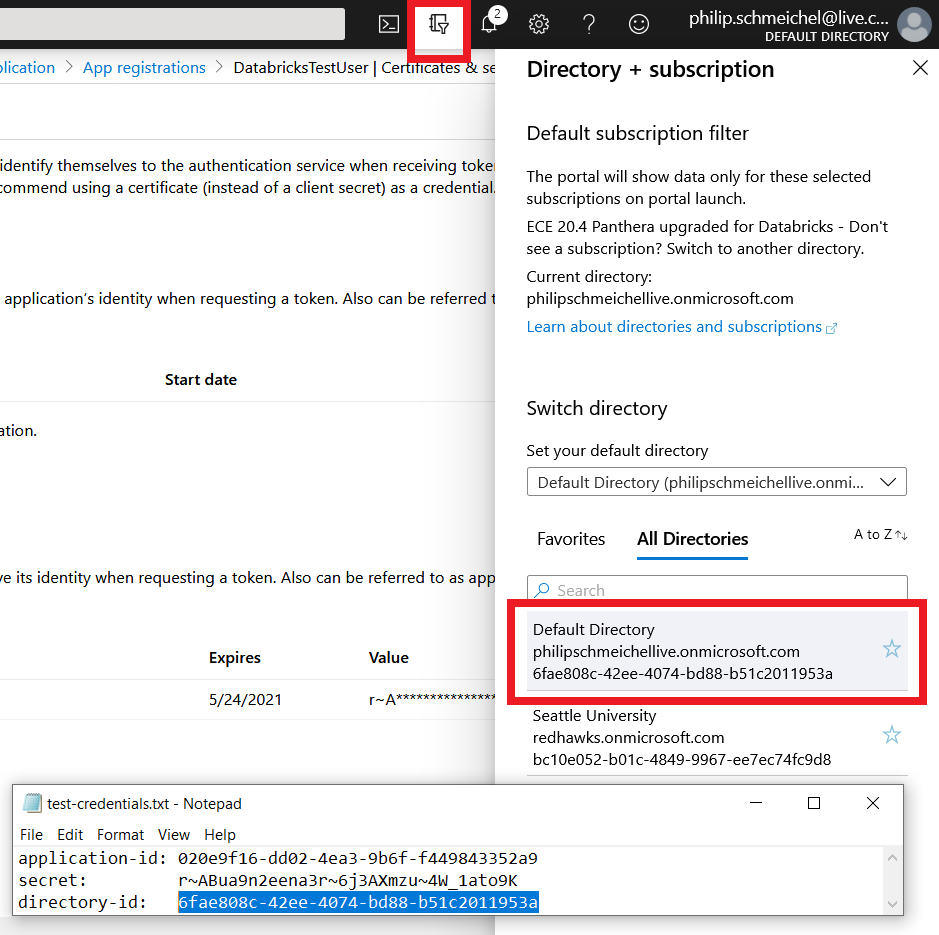


Figure 11. Directory ID recorded in Notepad

After copying the necessary info from the enterprise application, you will need to return to the storage account to give the enterprise application access to the storage container. In the storage account, select “Access control (IAM),” where you will adjust the enterprise application’s role to “Storage Blob Data Contributor.”

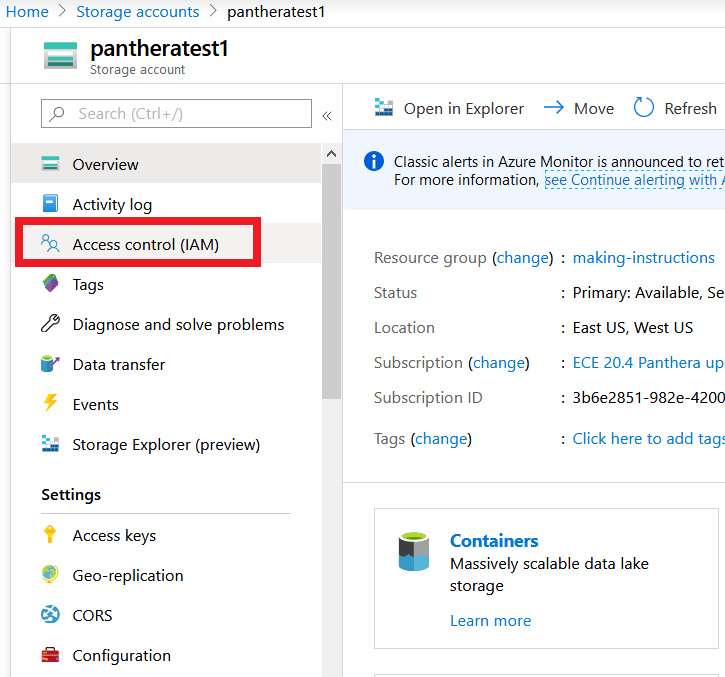


Figure 12. Access control location in storage account overview

In “Access control (IAM),” select “Role assignments” then “Add role assignment.” Select the Storage Blob Data Contributor role and type in the name of your enterprise application, then save the new role assignment.

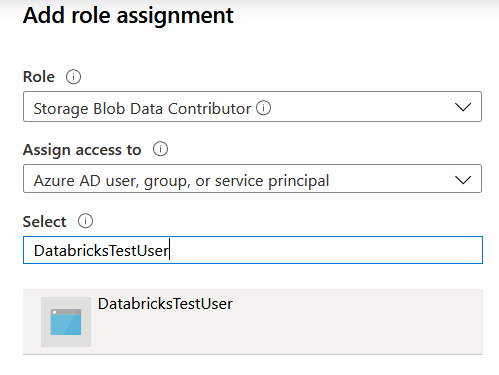


Figure 13. Assignment of Storage Blob Data Contributor role to enterprise application

## Azure Databricks Workspace

The last Azure resource to be created is the Azure Databricks Service, which is where the Spark notebook will be run on a “cluster,” or group of “worker” virtual machines. Now, your storage container should be ready to be used with Databricks, since these steps have been completed:

* Storage account created, with enterprise application assigned to the Storage Blob Data Contributor role
* Storage container created with images, templates, and *config.json* file stored inside as in a Recognition working directory on a local machine
* These fields copied for use in the Recognition Spark notebook in Databricks:
  + Enterprise application ID
  + Secret generated by enterprise application
  + Azure working directory ID (i.e. Default Directory ID)
  + Storage container name
  + Storage account name

To create the resource, search for “Databricks” in the Azure Portal, select “Add,” give the workspace a name, then “Review + Create.” You should then be able to access Databricks on the resource’s Overview page by clicking “Launch Workspace.”

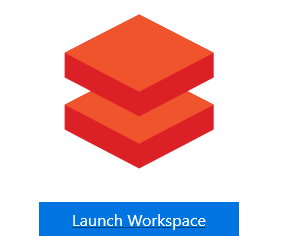


Figure 14. "Launch Workspace" icon

# Running the Recognition Spark Notebook in Databricks

Once in the Databricks workspace, you will see a screen with options for creating a notebook, cluster, dataset, and other Databricks features. In this part of the guide you will use the notebook *quick\_set-spark.ipynb* or *quick\_set-spark.py* to run the Recognition in the cloud. Begin by selecting “Clusters” on the left to set up the group of virtual machines that will run Recognition.

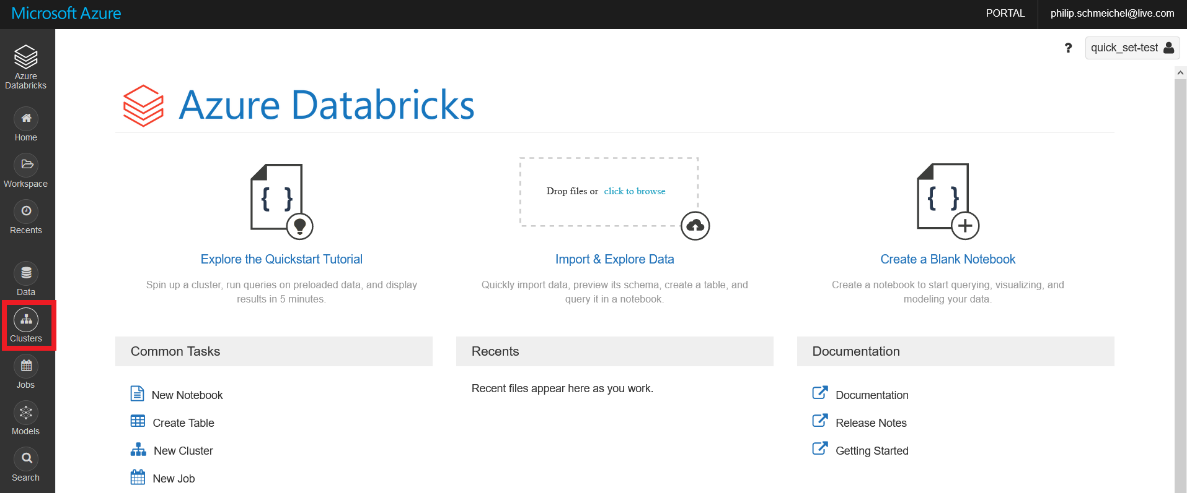


Figure 15. Databricks landing page, with "Clusters" highlighted

Create a cluster with the features outlined below or in the *quick\_set-spark* notebook. **To prevent high costs when running the cluster,** **uncheck “Enable autoscaling” and reduce both the time before termination due to activity and the number of workers** (NOTE: You may need to increase your quota for your selected worker type. This can be done by going back to the Azure Portal, searching for “Subscriptions,” and going to “Usage + quotas” under “Settings.” Then, as in Figure 1, filter to find your worker type and “Request increase” for that type.).

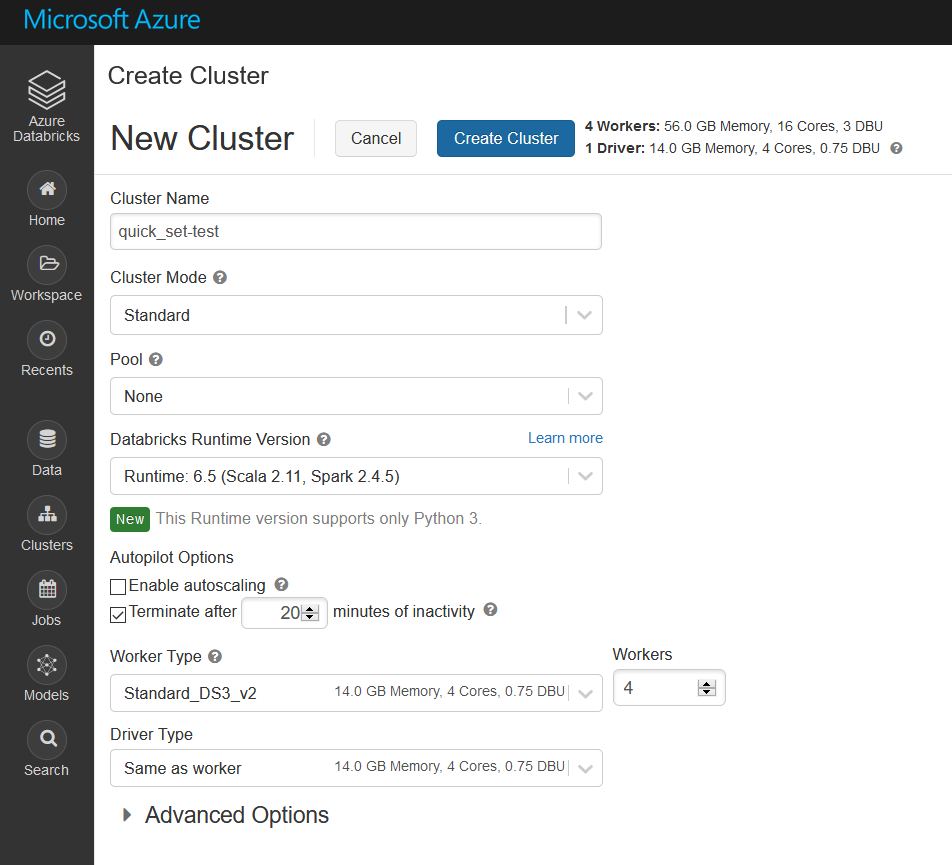


Figure 16. Cluster configuration for quick\_set tests

After creating your cluster, it will automatically start, showing a “Pending” state for up to 15 minutes. While it starts, install the following libraries by going into the cluster’s details. Click “Install New,” select “PyPI,” and install these 4 libraries: *IPython[all]*, *numpy*, *opencv-contrib-python-headless==3.4.2.17*, and *scikit-image*.

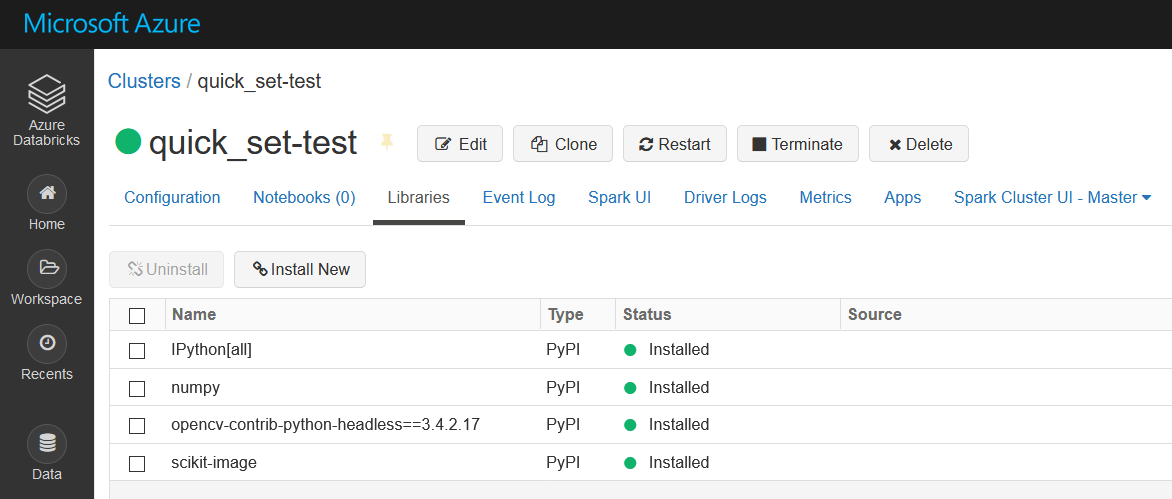


Figure 17. Libraries installed on the newly-made cluster

Next, create the notebook in the Azure Databricks launch page. Name it, make it a Python notebook, and attach it to the running cluster.

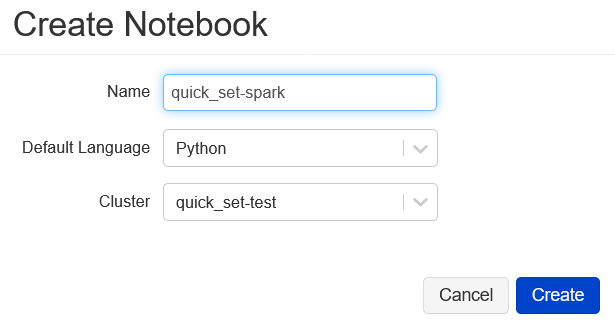


Figure 18. Notebook creation from launch page

At this point, you may open the python code for the *quick\_set-spark* notebook provided in ECE 20.4’s deliverable GitHub. Copy each command into a cell in the notebook you created in Azure Databricks.

You will then need to recall the fields you copied from the Azure portal for use in the first command in the notebook. The next figure shows which fields are adjusted to cause the mount to succeed. This command makes it so that you may treat the storage container as a directory in the Databricks File System. This means that the program will use *dbutils.fs* methods instead of *os* library methods in the Python code that works with paths and opening files. After the notebook runs, you will be able to access the “.CSV” file containing the score matrix in the Azure Portal (or Azure Storage Explorer) and download it to your local machine.

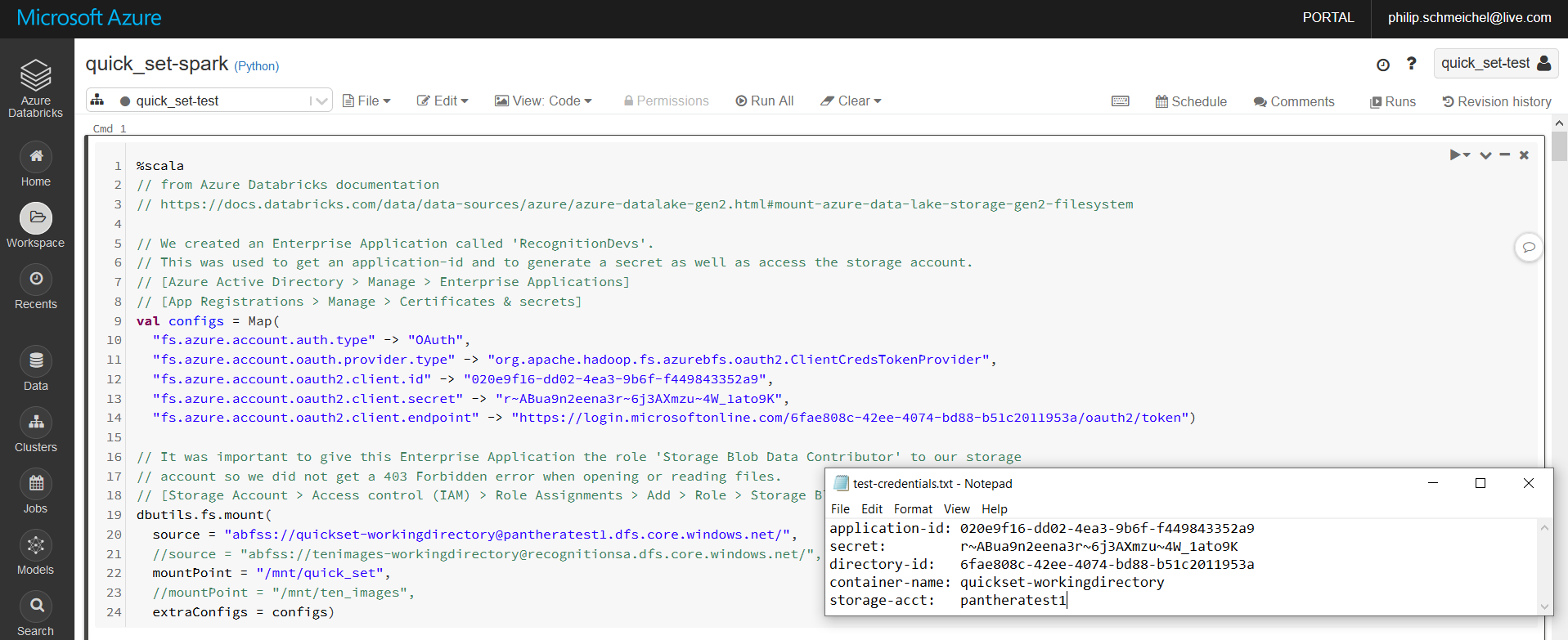


Figure 19. First command in the notebook, which takes care of mounting with credentials collected from the Azure Portal

Once the notebook is attached to the cluster and the mounting command works correctly, the entire notebook may be run using the “Run All” command. This will do the following in these commands:

* Mount the storage container to the Databricks File System
* Import the libraries used by the notebook
* Define the Recognition class and functions, implemented for the Spark environment used by Databricks
* Define an unused function for displaying images in a Spark notebook, written by Jonathan Scholtes
* Define the paths and *config.json* settings, as in local Recognition’s “main”
* Build the rec\_list of Recognition objects by accessing the mounted storage container’s images and templates
* Perform SIFT keypoint-matching and display the resulting score matrix
* Write the score matrix as a “.CSV” file to a specified folder inside the mounted storage container
* Unmount the storage container

Depending on the mount, the paths are customizable in the lines highlighted in the next figure but must follow these formats because of the different string syntax expected by *dbutils.fs.ls()* and *cv2.imread()*.



Figure 20. Path strings that are edited if the mount point is edited

**It is important to terminate the cluster once you are done running the notebook so that your subscription is not charged for resources that you are not using.** Before you close Databricks, make sure to select the “stop” icon that says “Terminate” and wait for the “State” to say “Terminated.”

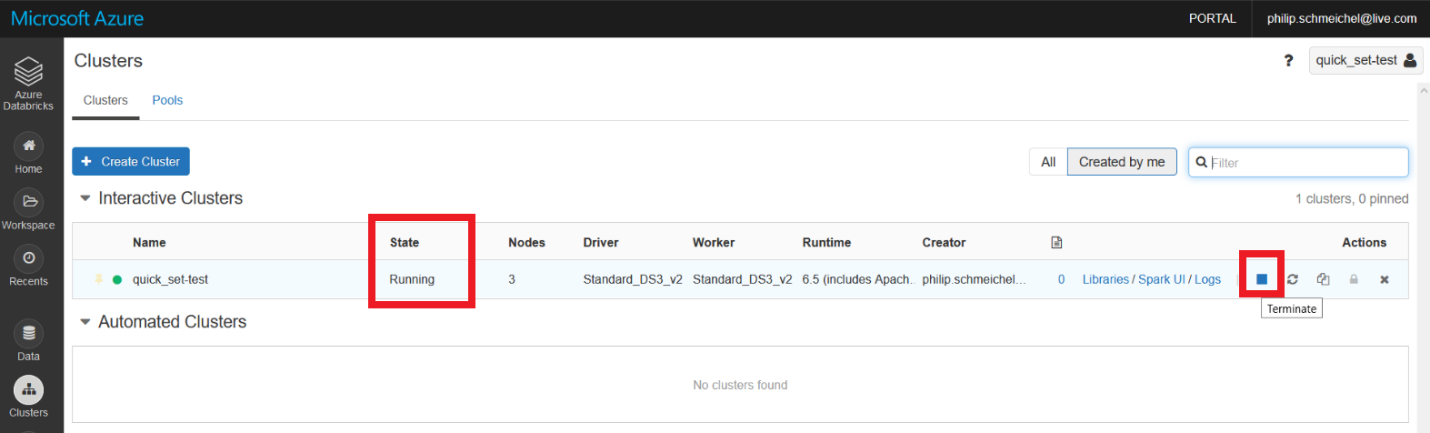


Figure 21. Termination of Databricks cluster, with the "State" and the "stop" icon highlighted

# Recognition Spark Notebook Output

The output of Recognition is the “score matrix,” a comma-separated values (.CSV) document that can easily be opened in Microsoft Excel. This indicates how many keypoint matches there are between each pair of images when compared after using the SIFT algorithm. It is a square matrix that has the length and width of the number of images in the dataset.

During the run of the Recognition notebook in Databricks, the score matrix is displayed underneath the command that performs Recognition’s *match\_multi* function.



Figure 22. Score matrix output displayed in Databricks

The score matrix is then written to a folder in the mounted storage container, by default into its own folder called “score\_matrix.” Because of the commands used to write the CSV in Databricks, the CSV file containing the score matrix has a much longer name beginning with “part-,” as if it were a partition waiting to be concatenated. The CSV can be downloaded and renamed “score\_matrix.csv” and it will serve the same function as it would in a regular local Recognition run, as input for a clustering method to guess the number of cats in this image set.

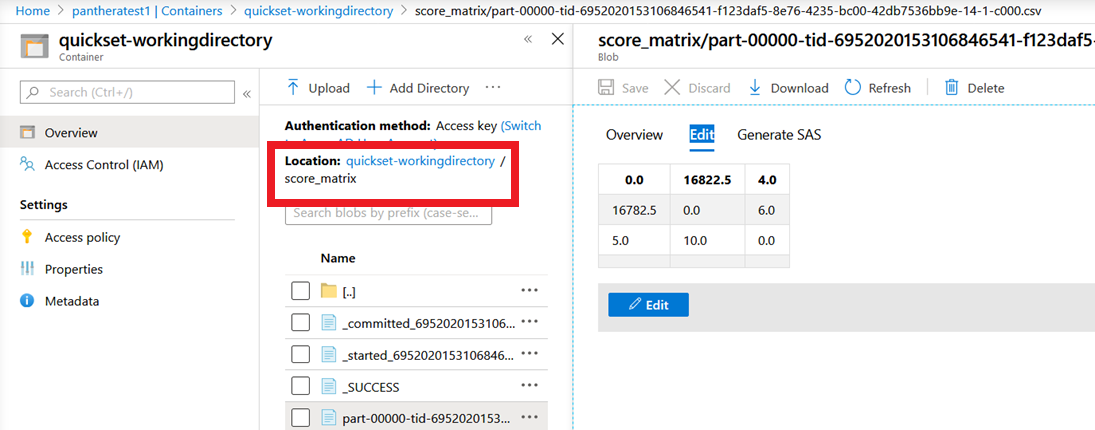


Figure 23. Score matrix accessed in storage container through the Azure Portal

# Progress of the Recognition Notebook

This section will outline what has been done while developing the Databricks notebook on Azure and what goals are for future development. The main issues currently are the slow runtime and implementation of the rest of the local version of Recognition’s functionality.

A bulk of the time working on the Recognition notebook was spent understanding the permissions required to initiate a mount of a storage account and troubleshooting routes to avoid “403 Forbidden” errors. Much of the rest of the time was spent adjusting the Recognition code and learning that *OpenCV* and *os* library methods would need to be adjusted or replaced to provide the same functionality in Databricks’ Spark environment.

The *OpenCV* method *imread()* in particular would not show an error if it was given a path that was not formatted correctly for the Databricks mount. Instead, it would return “None” to our image object. The next time this image object was used by an *OpenCV* method, an error would stop the command. The filenames were obtained by using *dbutils.fs.ls()* instead of *os.list()*. Both of these errors were resolved in the *init\_recognition* and *add\_templates* functions.

It takes the Databricks notebook as long or longer than a local run of Recognition to generate a score matrix for both the image datasets quick\_set (2.5 minutes) and ten\_images (over 45 minutes). We believe that not all the worker machines are being used by the cluster to do the SIFT keypoint detection because of the multithreading version of Recognition being used in this notebook. From reading online, Ben Weber has posted solutions for parallelizing Spark workloads. In his writeup, he suggested that possibly only the “driver” node (that controls the worker machines) may be doing the computations, and he gives examples of multiprocessing in Python using Spark.

In early 2019, Ross Pitman had a version of Recognition that used multiprocessing. Team ECE 20.4 would suggest that other teams get a hold of this code and learn more about multiprocessing in Spark.

The rest of Recognition’s functionality, such as image enhancement, template generation with the Mask R-CNN, and clustering functions implemented by ECE 20.4 still need to be implemented in Spark. The goal of this year’s team with cloud computing on Azure Databricks was to prove that the service could be used for faster tests, regardless of platform or cloud service provider. There are still further paths to learn about, such as pipelining the functions of the different modules of Recognition, possibly including automatic re-training of the Mask R-CNN. This would push Panthera’s Project Spot Check further into a focus on cloud computing and Internet-of-Things, alongside heavier machine-learning problems.

All of these developments would continue toward the goal of testing Recognition’s performance with very large datasets. The score matrices obtained from running these large tests will aid in the validation of Recognition’s accuracy in identifying snow leopards.

# References

Special thanks to Devin DeWitt, member of team ECE 19.7, whose guidance was invaluable in understanding the “why” and “how” of Azure applications for ADLS Gen 2 storage, enterprise applications, and Databricks concepts.

Azure/Databricks Documentation

<https://docs.microsoft.com/en-us/azure/guides/developer/azure-developer-guide>

<https://docs.microsoft.com/en-us/learn/paths/azure-fundamentals/>

<https://docs.microsoft.com/en-us/azure/storage/blobs/data-lake-storage-introduction>

[https://docs.databricks.com/data/data-sources/azure/azure-datalake-gen2.html#mount-azure-data-lake-storage-gen2-filesystem](https://docs.databricks.com/data/data-sources/azure/azure-datalake-gen2.html" \l "mount-azure-data-lake-storage-gen2-filesystem)

<https://docs.databricks.com/data/data-sources/read-csv.html>

Code and tips from other sources

Jonathan Scholtes: <https://stochasticcoder.com/2018/06/06/python-image-processing-on-azure-databricks-part-1-opencv-image-compare/>

Adrian Rosebrock: <https://www.pyimagesearch.com/2016/12/26/opencv-resolving-nonetype-errors/>

Ben Weber: <https://towardsdatascience.com/3-methods-for-parallelization-in-spark-6a1a4333b473>

Stack Overflow

<https://stackoverflow.com/questions/51932783/how-to-loop-through-azure-datalake-store-files-in-azure-databricks>

<https://stackoverflow.com/questions/38696383/accessing-images-in-spark-using-python-and-opencv>

<https://stackoverflow.com/questions/31674530/write-single-csv-file-using-spark-csv>

<https://stackoverflow.com/questions/53537602/convert-a-numpy-array-to-a-dataframe-in-pyspark-to-export-as-csv>